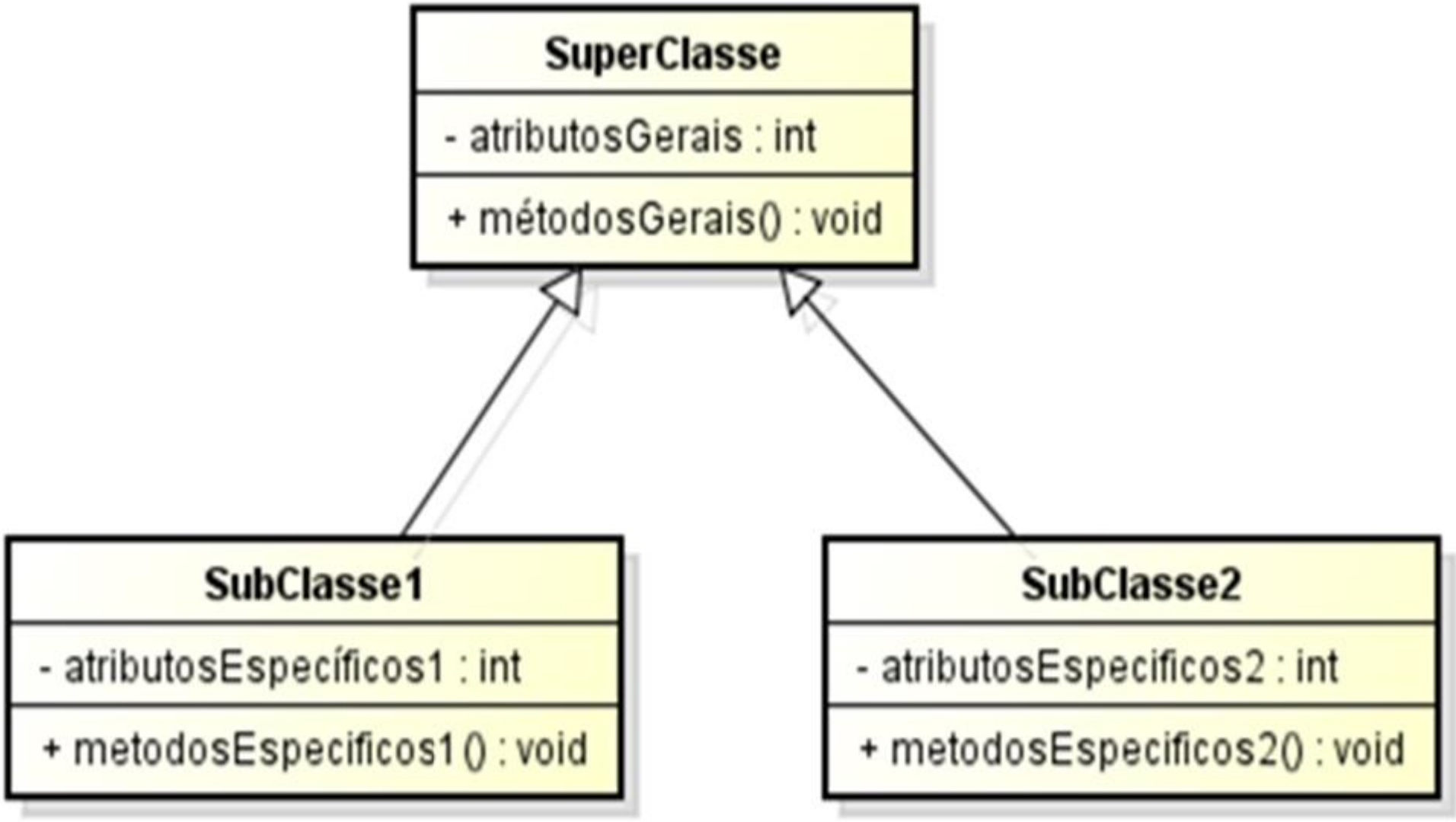
Programação Orientada a Objetos

Aula de Exercícios sobre Herança

Professores: Dirson, Nádia e Juliana

Revisão de Conceitos Básicos da Aula Anterior sobre Herança

Programação Orientada a Objetos
Recordando alguns tópicos básicos da aula anterior



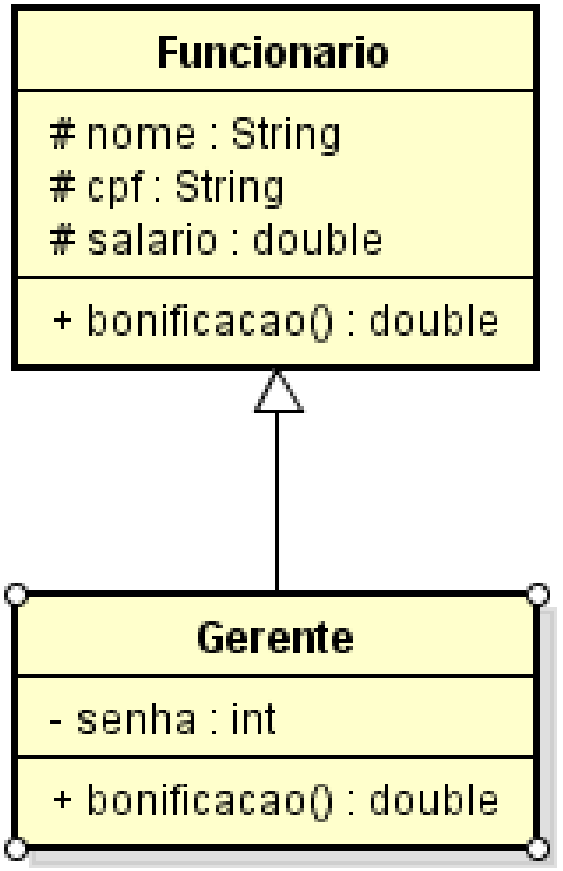
Programação Orientada a Objetos

Vamos recordar um exemplo da aula anterior sobre o uso de herança:

- Uma classe Funcionário tem nome, cpf e salário. Tem também um método que calcula bonificação de 10% do salário.
- Todo Gerente, é um Funcionário e como tem acesso restrito a alguns lugares, ele tem uma senha.
- O cálculo da bonificação do Gerente é diferente, sendo 15% do salário.

Programação Orientada a Objetos

Diagrama UML do exemplo do slide anterior (exemplo 1)



Programação Orientada a Objetos

Implementação em Java do Exemplo 1, sem o método construtor explícito.

```
public class Funcionario {
        protected String nome;
        protected String cpf;
        protected double salario;

public double bonificacao(){
        double b = salario * 0.10;
        return b;
}

public double getSalario() {
        return salario;
}

public void setSalario(double salario) {
        this.salario = salario;
}

}
```

```
public class Gerente extends Funcionario{
        private int senha;

public double bonificacao(){
        double b = salario * 0.15;
        return b;
}

}
```

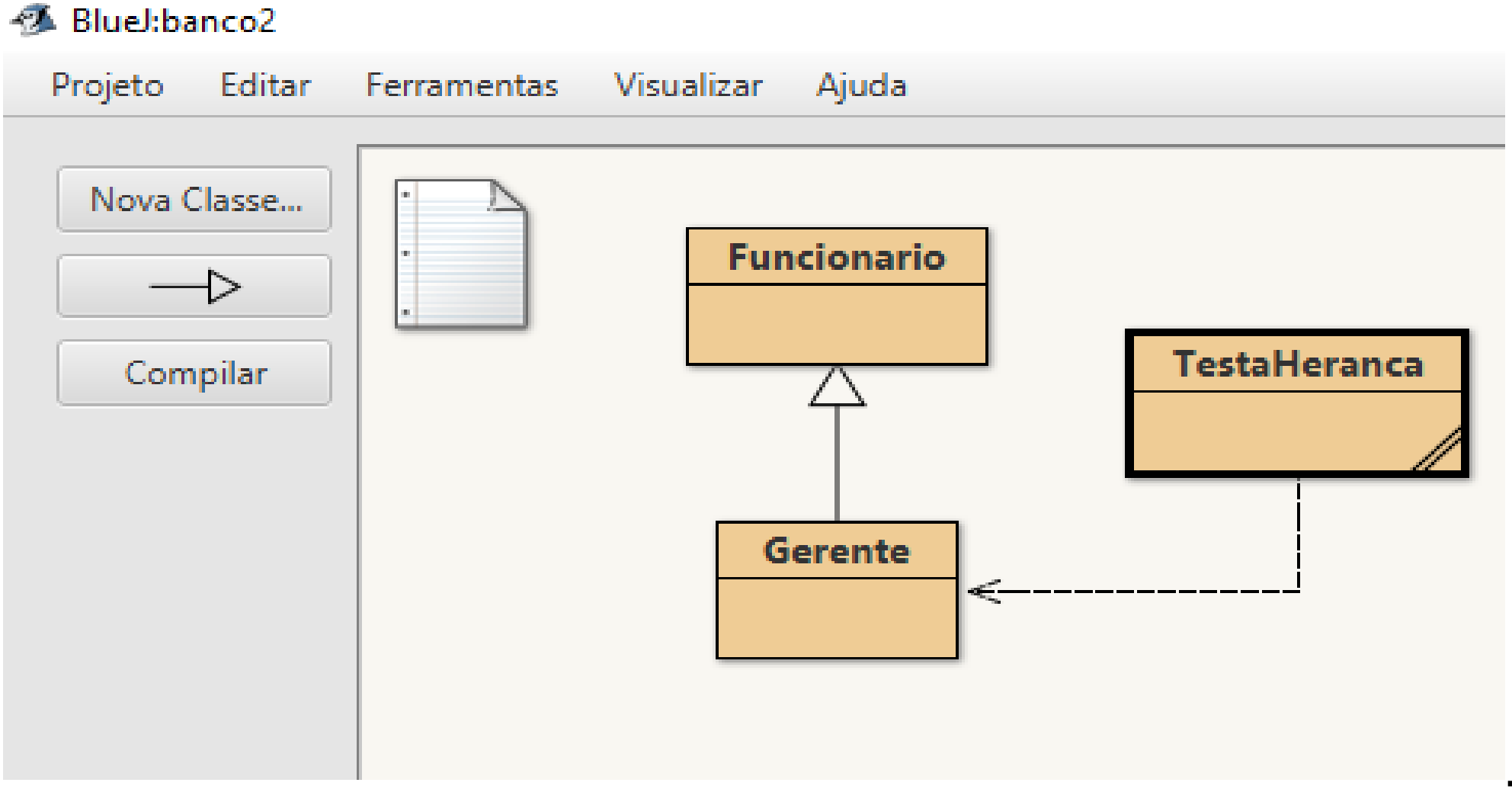
```
public class TestaHeranca {

public static void main(String [] args){
        Gerente g = new Gerente();
        g.setSalario(3000);
        System.out.println("A bonificacao é:
" +     g.bonificacao());
}

}
```

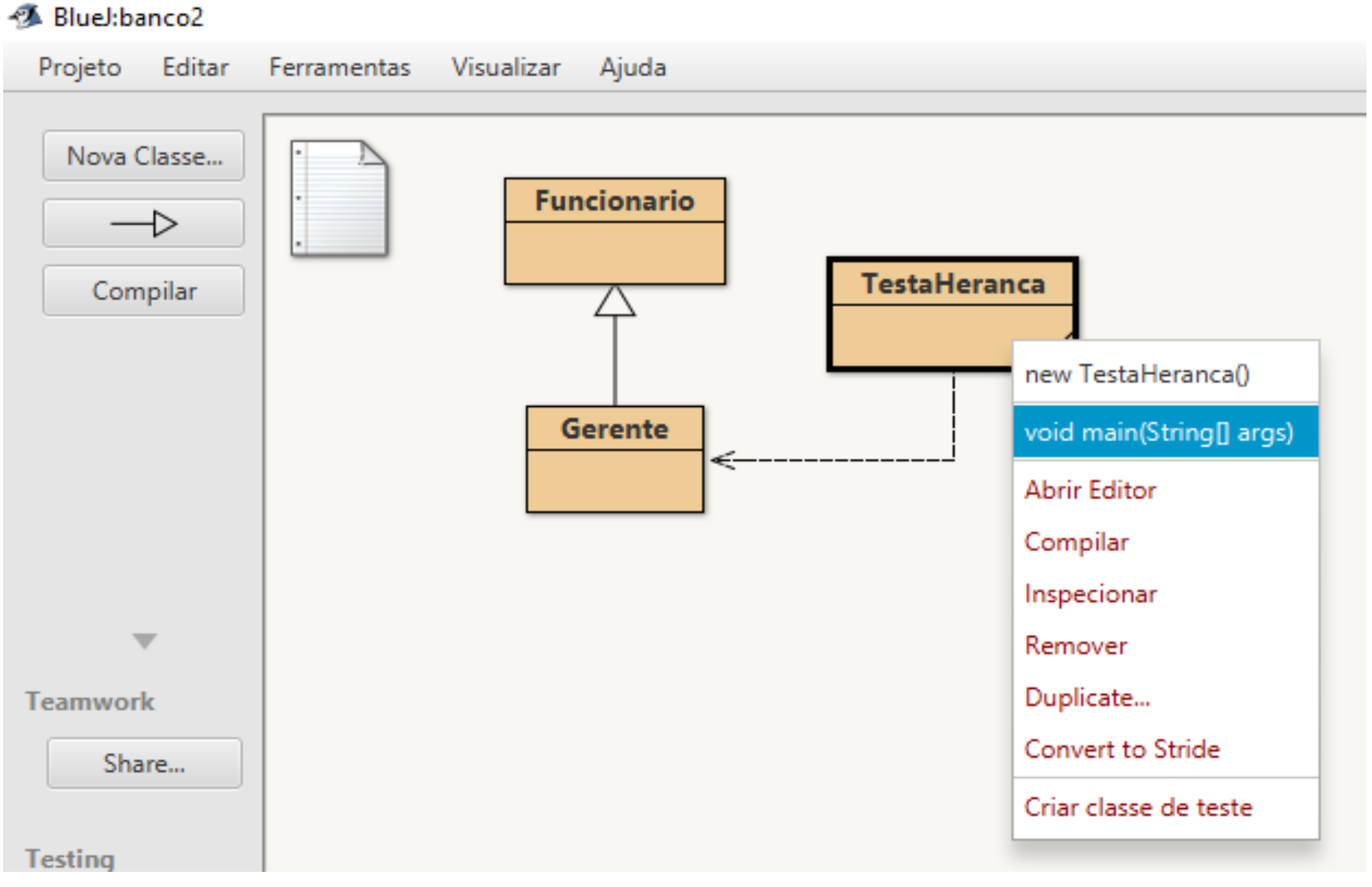
Programação Orientada a Objetos

Diagrama de Classe do exemplo 1



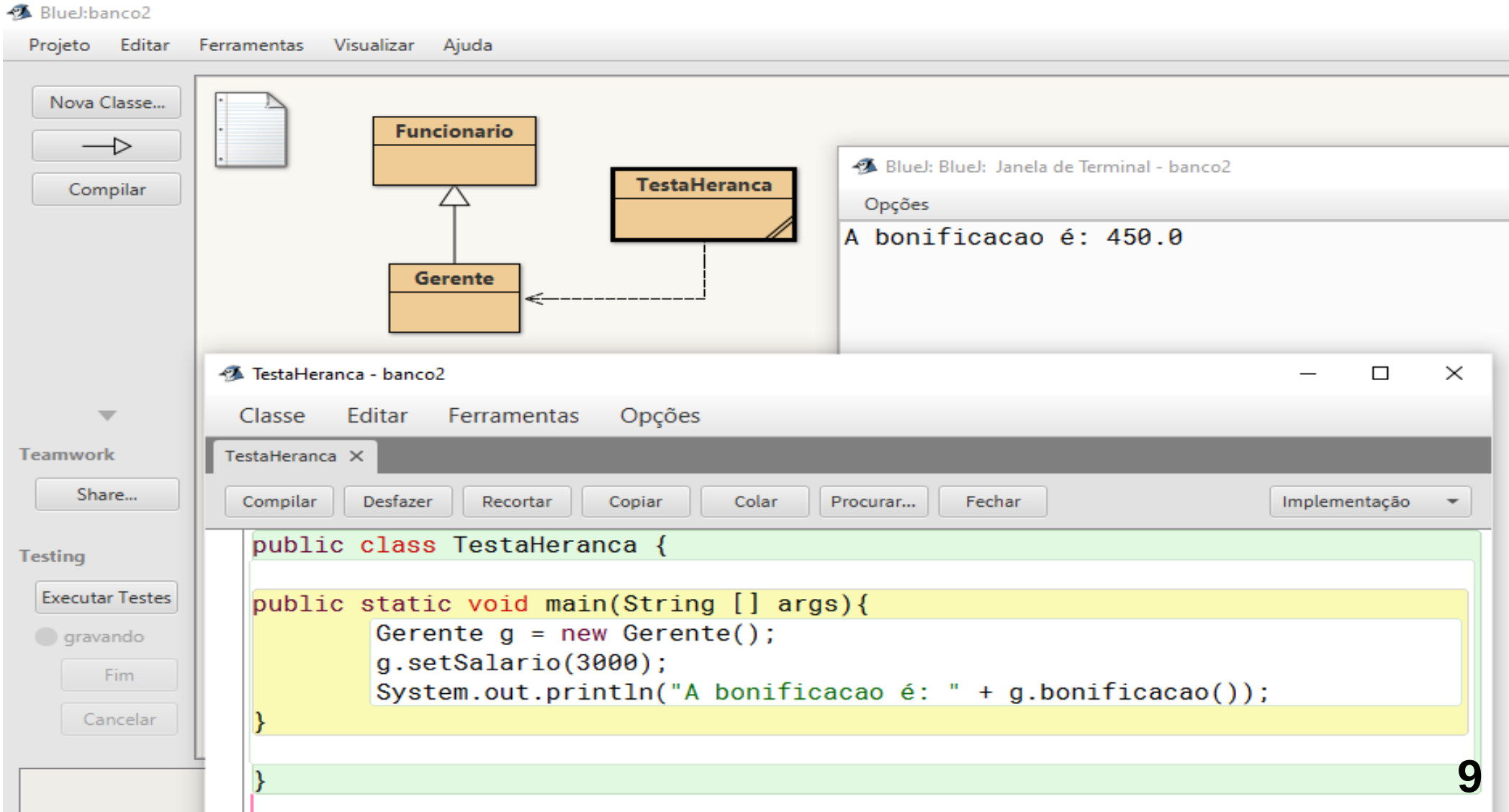
Programação Orientada a Objetos

Testando o método main do exemplo 1 no BlueJ



Programação Orientada a Objetos

Execução do Exemplo 1 no BlueJ



Programação Orientada a Objetos
Slide da aula anterior de Herança

Como identificar e modelar a herança (recordando a aula anterior ...)

- Identificar as entidades importantes do contexto;
- Identificar as características (dados) e comportamentos (operações) de cada uma;
- Identificar características e comportamentos comuns (gerais) nas entidades;
- Identificar características e comportamentos específicos em cada entidades;
- Agrupar características e comportamentos comuns em uma superclasse (classe pai);
- Manter características e comportamentos específicos em cada classe;

Programação Orientada a Objetos

Exemplo da aula anterior usando o método construtor:

```
public class Funcionario {
        protected String nome;
        protected String cpf;
        protected double salario;

Funcionario(String nome,String cpf){
        this.nome = nome;
        this.cpf = cpf;
}
...
}
```

```
public class Gerente extends Funcionario{
        private int senha;

Gerente (String nome,String cpf, int senha){
        super(nome,cpf);
        This.senha = senha;
}


}
```

```
public class TestaHeranca {

public static void main(String [] args){
        Funcionario f = new Funcionario(“Jose”,”23435678999”);
        Gerente g = new Gerente(“Paulo”,”9949595595”,3344);
        System.out.println("Funcionario: " + f.getNome());
    System.out.println("Gerente: " + g.getNome());
}
}
```

Programação Orientada a Objetos
Recordando a aula anterior Herança

Métodos construtores nas subclasses

- O construtor de uma subclasse **sempre** chama o construtor de sua superclasse, mesmo que a chamada não seja explícita.

- Se a chamada não for explícita (através da palavra-chave **`super`**) o construtor da subclasse tentará chamar o construtor vazio (sem argumentos) da superclasse – e se ele não estiver definido, ocorrerá um erro de compilação;

- Se uma classe não possui um construtor vazio (sem argumentos) e possui um construtor com argumentos, as classes herdeiras deverão obrigatoriamente chamar o construtor com argumentos da classe ancestral (este é um tipo de erro que geralmente causa muita confusão).

Resolvendo Exercícios da Lista Sobre Herança

Programação Orientada a Objetos
Exercício 5.1 da Lista Exercícios - POO unidade 1

Enunciado:

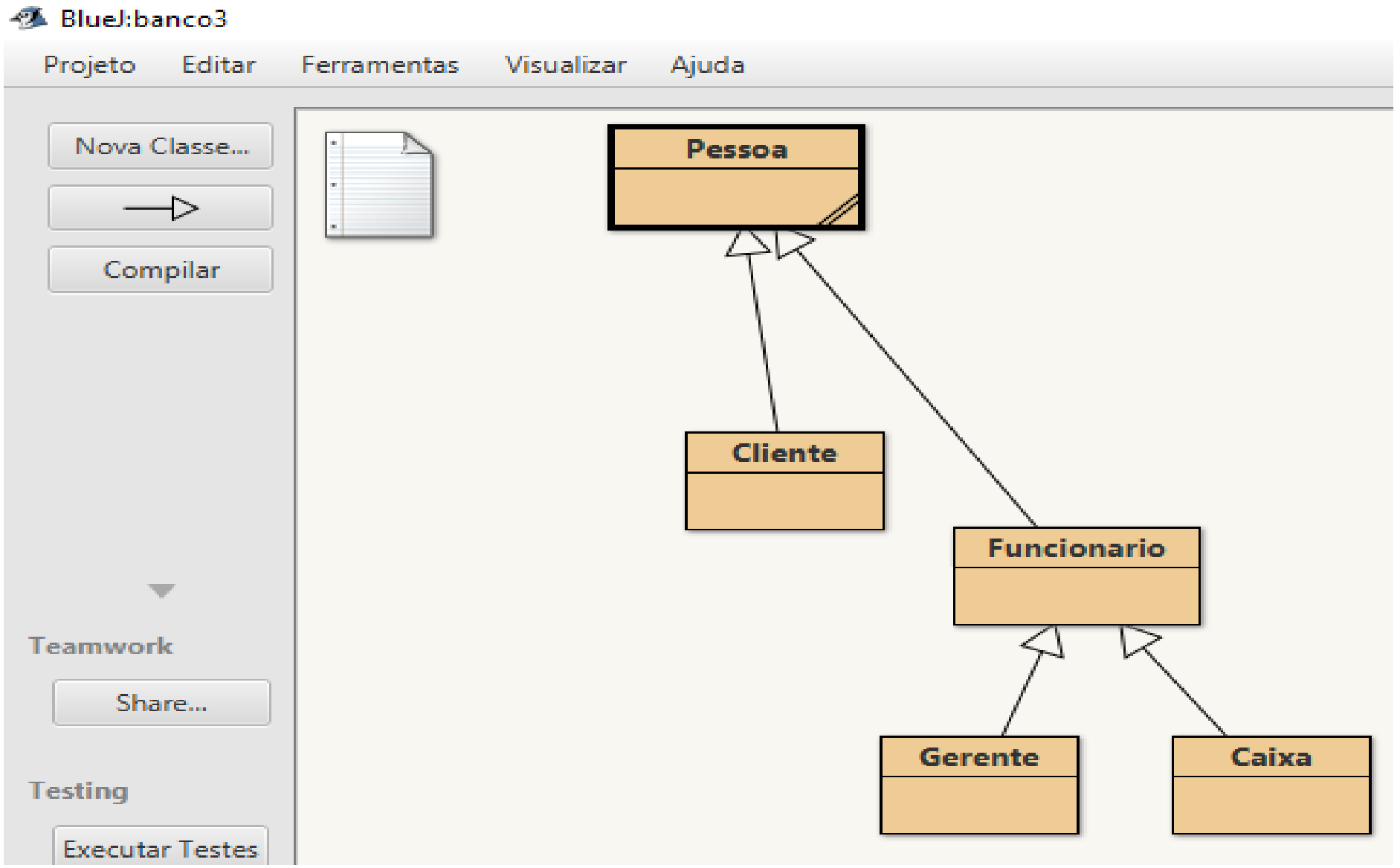
Considere a seguinte situação:

Um banco deseja controlar os dados de todos os tipos de pessoas que se relacionam com os seus sistemas, tanto **funcionários** quanto **clientes**. Para isso, foram levantadas algumas informações sobre características de cada tipo de **pessoa**.

- Um caixa de banco é um funcionário e tem como características: **nome**, **telefone**, **matricula, salário, horário**;
- Um gerente é um funcionário e tem como características: **nome**, **telefone**, **matrícula, salário, bonificação (fixa mensal) e um tipo (PF para gerente de pessoa física ou PJ para jurídica);**
- Um cliente tem como características: **nome**, **telefone**, **idade, cpf, status ("A" – ativo, "I" - inativo)**

5.1 - Escreva as classes, com seus atributos, necessárias para representar o enunciado usando o conceito de herança e as boas práticas (atributos protegidos e métodos públicos).

Possível solução do Exercício 5.1 da Lista Exercícios - POO unidade 1 no Bluej. Diagrama Classe no BlueJ



Programação Orientada a Objetos Exercício 5.1 da Lista Exercícios - POO unidade 1 – Classe Pessoa

```
public class Pessoa {
        protected String nome;
        protected String telefone;
 /**
    * Construtor que recebe os atributos da pessoa.
    *
    */
   public Pessoa(String nome, String telefone) {
       this.nome = nome;
       this.telefone = telefone;
   }
}
```

Nesta solução será usado o método construtor feito pelo próprio programador . Desde que a solução do estudante atenda o enunciado do exercício é possível ter mais de uma solução válida e logicamente equivalente a descrita neste slide.

Programação Orientada a Objetos Exercício 5.1 da Lista Exercícios - POO unidade 1 – Classe Cliente

```
public class Cliente extends Pessoa{
    private int idade;
    private String cpf;
    private char status;

/**
     * Construtor que recebe os atributos do Cliente, inclusive os herdados.
     *
     */

  public Cliente (String nome, String telefone, int idade, String cpf, char
status) {
        super(nome, telefone);
        this.idade = idade;
        this.cpf = cpf;
        this.status = status; }
    }
```

Programação Orientada a Objetos Exercício 5.1 da Lista Exercícios - POO unidade 1 – Classe Funcionario

```
public class Funcionario extends Pessoa{
    protected String matricula;
    protected float salario;
  /**
     * Construtor que recebe os atributos do Caixa, inclusive os herdados.
     *
     */
     public Funcionario (String nome, String telefone, String matricula, float salario)
{       super(nome, telefone);
        this.matricula = matricula;
        this.salario = salario;
    }
    public double getSalario() { return salario; }
public void setSalario(float salario) { this.salario = salario;}
}
```

Programação Orientada a Objetos Exercício 5.1 da Lista Exercícios - POO unidade 1 – Classe Gerente

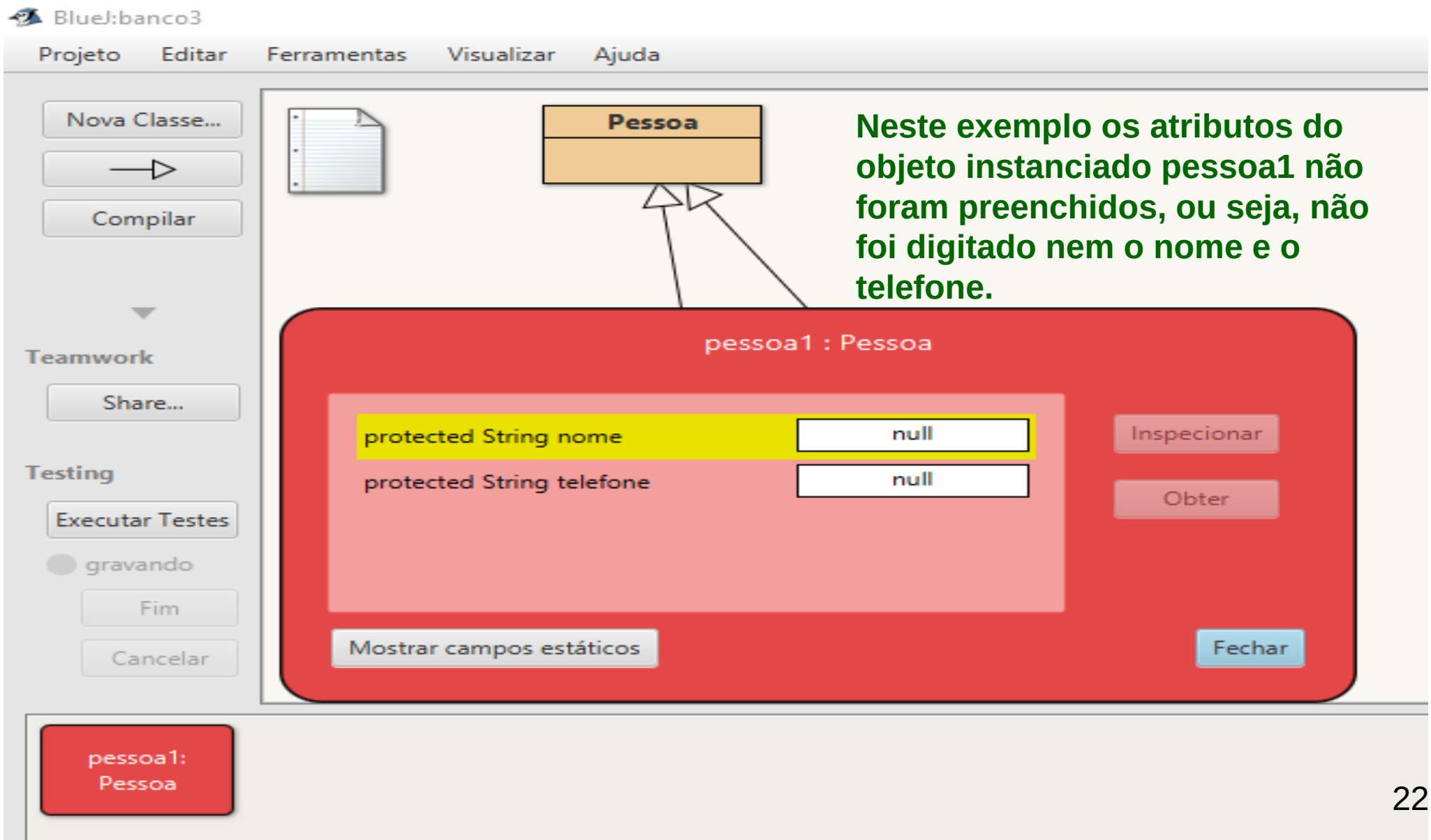
```
public class Gerente extends Funcionario{
        private float bonificacao;
        private String tipo;
/**
 Construtor que recebe os atributos do Gerente, inclusive os herdados.
  */
 public Gerente (String nome, String telefone, String matricula, float salario, float bonificacao,  String tipo) {
      super(nome, telefone, matricula, salario);
       this.bonificacao = bonificacao;
       this.tipo = tipo;
   }   }
```

Programação Orientada a Objetos Exercício 5.1 da Lista Exercícios - POO unidade 1 – Classe Caixa

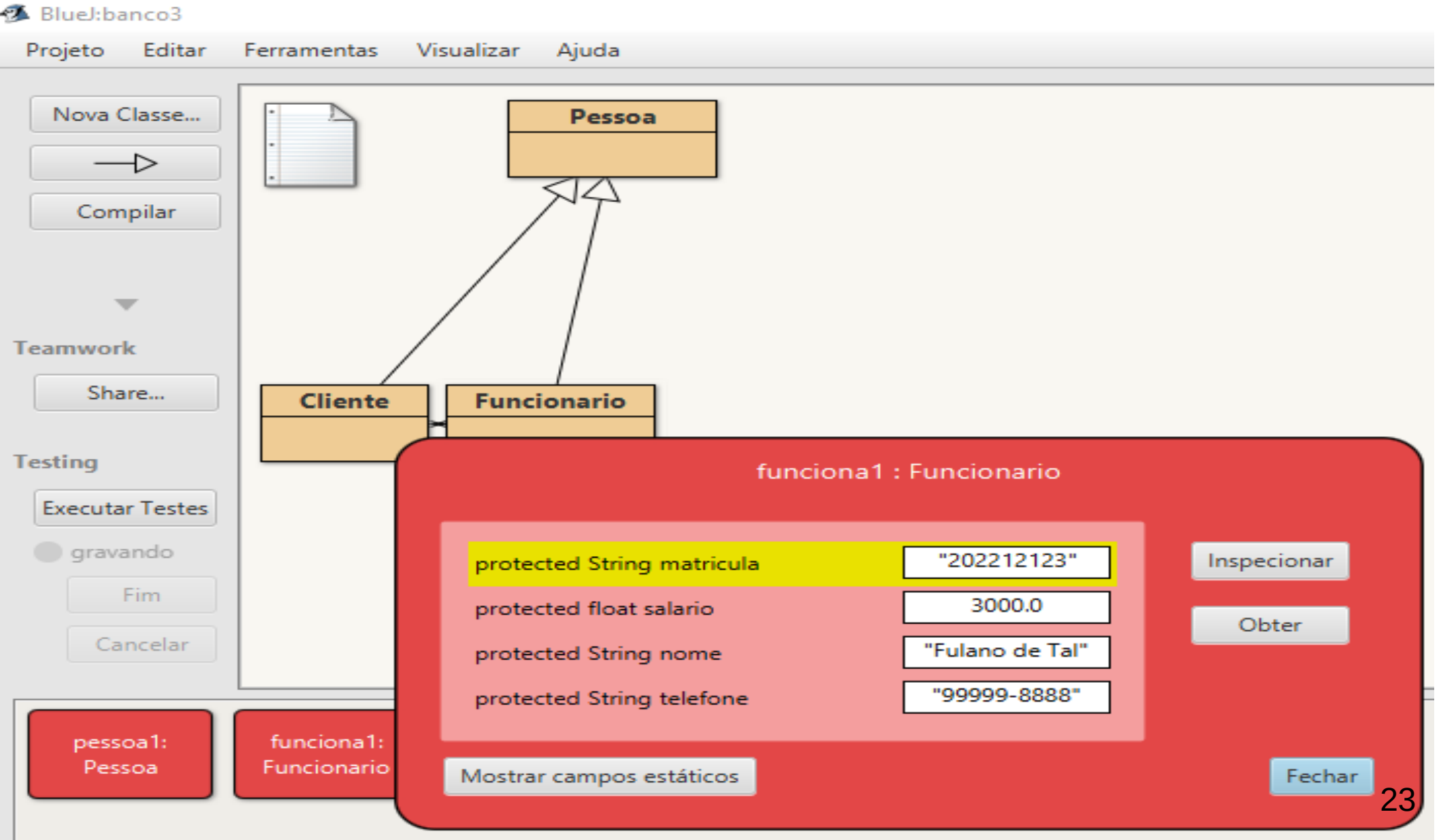
```
public class Caixa extends Funcionario{
    private String horario;
  /**
  Construtor que recebe os atributos do Caixa, inclusive os herdados.
    */
  public Caixa (String nome, String telefone, String matricula,
float salario,  String horario) {
      super(nome, telefone, matricula, salario);
      this.horario = horario;
    }
}
```

Testando o método construtor de algumas classe e um dos métodos do tipo get e set usando a ferramenta BlueJ do exercício 5.1

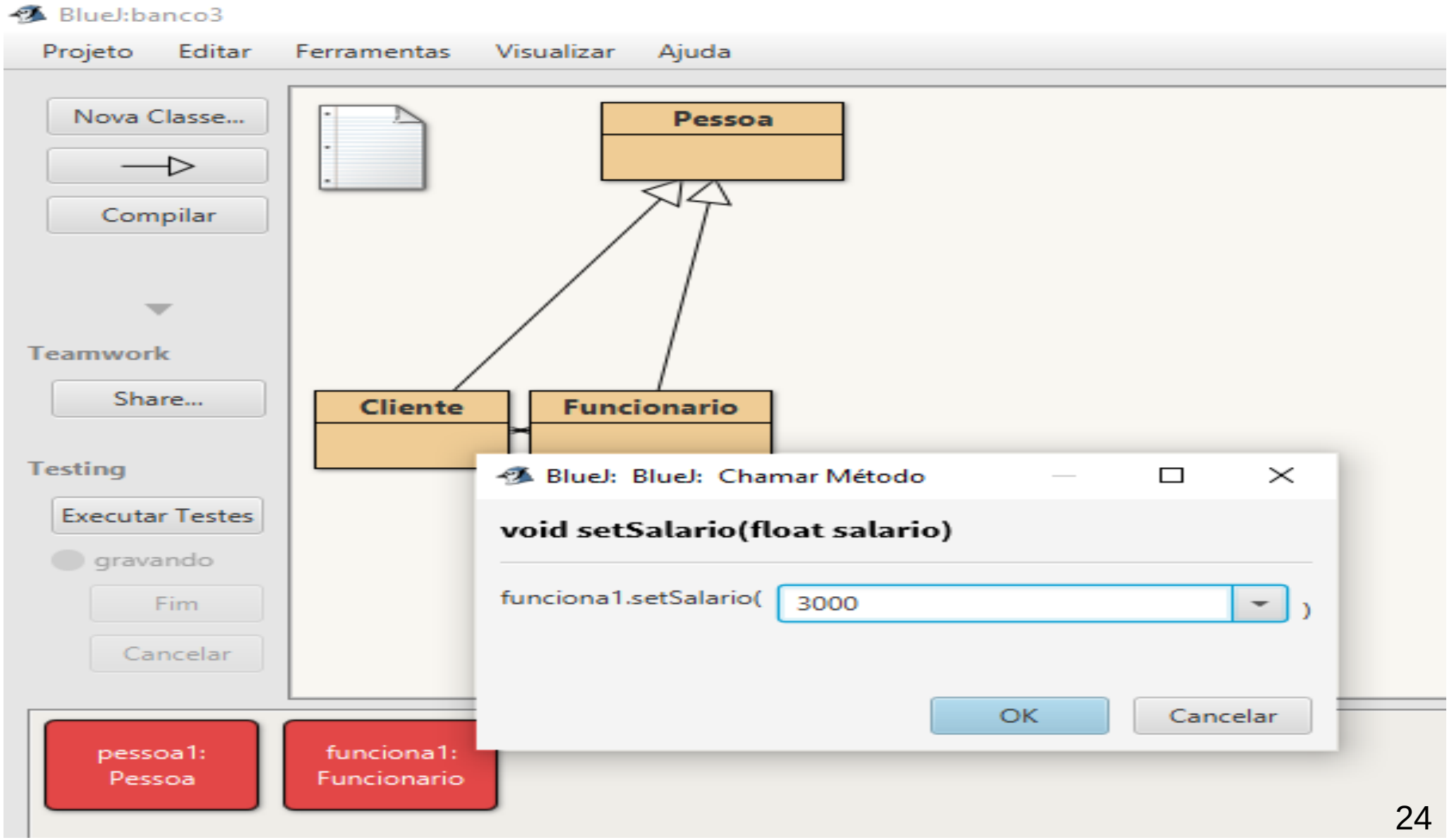
Exercício 5.1 da Lista Exercícios - POO unidade 1 – Exemplo de instanciamento de um objeto da classe Pessoa usando o BlueJ



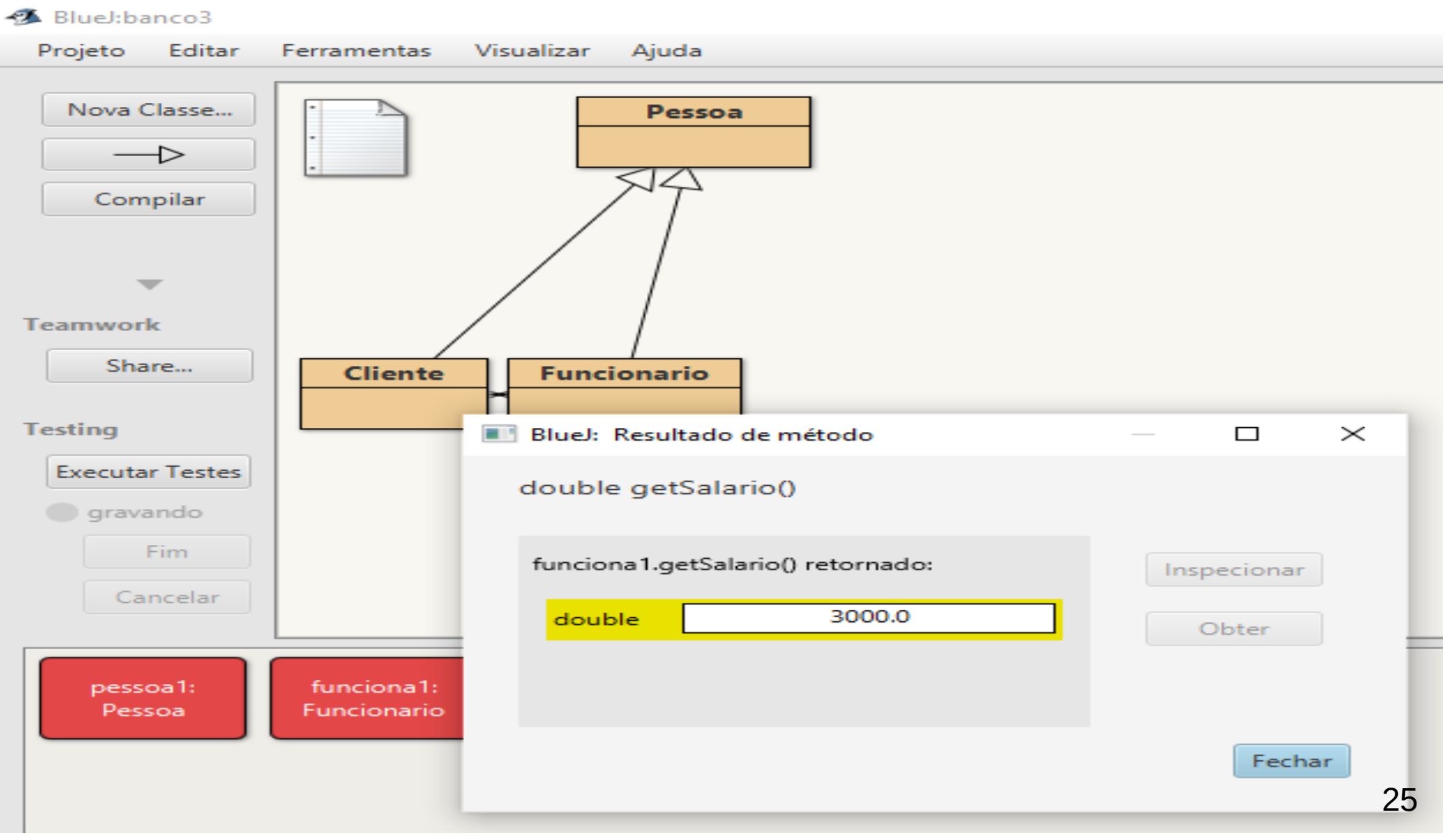
Exercício 5.1 da Lista Exercícios - POO unidade 1 – Exemplo de instanciamento de um objeto da classe Funcionario usando o BlueJ



Exercício 5.1 da Lista Exercícios - POO unidade 1 – Testando a chamada do método do tipo set usando o BlueJ



Exercício 5.1 da Lista Exercícios - POO unidade 1 – Testando a chamada do método do tipo get usando o BlueJ



Resolução parcial do exercício 5.2 da 01-Lista Exercicios - POO unidade 1

Programação Orientada a Objetos Exercício 5.2 da Lista Exercícios - POO unidade 1. Solução Parcial

Enunciado:

5.2 – Escreva os métodos das classes de acordo com os seguintes requisitos:

- **Todos os funcionários do banco podem ter aumento de salário sendo que este é aplicado através da chamada de um método que recebe como parâmetro um percentual e atualiza o salário somando o salário atual com esse percentual;**
- **O cliente deve ter um método para mostrar seus dados concatenados com a informação se ativo ou inativo. Ex: "Maria – Tel:2345-6697 – idade: 20 anos – cpf: 8889989898 - ativo"**

- **Obs.: Os demais itens do exercícios 5.2 e a classe TestaHeranca.java com o método main serão feitos pelos estudante, bem como toda a lista que deve ser submetida até o dia 14/02/2022.**

Programação Orientada a Objetos Exercício 5.2 da Lista Exercícios - POO unidade 1. Solução Parcial

Solução:

5.2 – Escreva os métodos das classes de acordo com os seguintes requisitos:

- **Todos os funcionários do banco podem ter aumento de salário sendo que este é aplicado através da chamada de um método que recebe como parâmetro um percentual e atualiza o salário somando o salário.**

```
public float novo_salario(float percentual){
        salario = salario + (salario * percentual);
        return salario;
}
```

Programação Orientada a Objetos Exercício 5.2 da Lista Exercícios - POO unidade 1. Solução Parcial

Solução:

5.2 – Escreva os métodos das classes de acordo com os seguintes requisitos:

- **O cliente deve ter um método para mostrar seus dados concatenados com a informação se ativo ou inativo. Ex: “Maria – Tel:2345-6697 – idade: 20 anos – cpf: 8889989898 - ativo”**

```
public void concatena_cliente(String nome, String telefone, int idade, String cpf, char status){
   String nome_status;
   if (status == 'A')
       nome_status = "ativo";
       else if (status == 'I')
           nome_status = "inativo";
           else nome_status = "Não informado, somente A ou I como entrada";

   System.out.println("\n" + this.nome + " – Tel:" + this.telefone + " – idade: " + this.idade);
   System.out.println(" anos – cpf: "  + this.cpf + " - " + nome_status);
}
```

Programação Orientada a Objetos Exercício 5.2 da Lista Exercícios - POO unidade 1. Solução Parcial

Teste dos dois métodos no BlueJ para um salário inicial de 3000 reais e os dados do enunciado do exemplo, a saber,

Ex: “Maria – Tel:2345-6697 – idade: 20 anos – cpf: 8889989898 - ativo”

BlueJ: BlueJ: Janela de Terminal - banco3

Opções

```
O salario do funcionario é: 3000.0
Qual o percentual de aumento em porcentagem?

15

O novo salario é: 3450.0

Maria – Tel:2345-6697 – idade: 20
 anos – cpf: 8889989898 - ativo
```